The Sieve of Eratosthenes

埃拉托塞尼篩法

埃拉托塞尼

- Eratosthenes
- Ερατοσθενης (276B.C. ~ 195 B.C.)
- 古希臘數學家、亞歷山大圖書館館長

- Create an array with all elements initialized to 1 (true).
  Array elements with prime subscripts will remain 1.
  All other array elements will eventually be set to zero.
  You will ignore elements 0 and 1 in this exercise.
- Starting with array subscript 2, every time an array element is found whose value is 1, loop through the remainder of the array and set to zero every element whose subscript is a multiple of the subscript for the element with value 1.
  For array subscript 2, all elements beyond 2 in the array that are multiples of 2 will be set to zero (subscripts 4, 6, 8, 10, etc.); for array subscript 3, all elements beyond 3 in the array that are multiples of 3 will be set to zero (subscripts 6, 9, 12, 15, etc.); and so on.
- When this process is complete, the array elements that are still set to one indicate that the subscript is a prime number.

The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0

The Sieve of Eratosthenes

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0

```
bool primes[ 1000000001 ];

int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;

   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )








   return numPrimes;
}
```

```
bool primes[ 1000000001 ];

int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;

   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )
      if( primes[ i ] )
      {
         numPrimes++;


      }

   return numPrimes;
}
```

```
bool primes[ 1000000001 ];

int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;

   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 2 * i; j <= number; j += i )
            primes[ j ] = false;
      }

   return numPrimes;
}
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

```
int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;
   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 2 * i; j <= number; j += i )
            primes[ j ] = false;
      }
   return numPrimes;
}
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

```
int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;
   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 2 * i; j <= number; j += i )
            primes[ j ] = false;
      }
   return numPrimes;
}
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0	0	0

```
int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;
   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 2 * i; j <= number; j += i )
            primes[ j ] = false;
      }
   return numPrimes;
}
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	0	1	0	1	0	0	0	1	0	1	0	0	0	1	0	1	0

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40
0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0

```
int eratosthenes( int number )
{
   for( int i = 2; i <= number; i++ )
      primes[ i ] = true;
   int numPrimes = 0;
   for( int i = 2; i <= number; i++ )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 2 * i; j <= number; j += i )
            primes[ j ] = false;
      }
   return numPrimes;
}
```

```
bool primes[ 1000000001 ];

int eratosthenes( int number )
{
   for( int i = 3; i <= number; i += 2 )
      primes[ i ] = true;

   int numPrimes = 1;
   for( int i = 3; i <= number; i += 2 )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 3 * i; j <= number; j += i + i )
            primes[ j ] = false;
      }

   return numPrimes;
}
```

```
bool primes[ 1000000001 ];

int eratosthenes( int n )
{
   for( int i = 2; i <= n; i++ )
      primes[ i ] = true;

   int squareRoot = static_cast< int >( sqrt( static_cast< double >( n ) ) );
   int numPrimes = 0;

   for( int i = 2; i <= squareRoot; i++ )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 2 * i; j <= n; j += i )
            primes[ j ] = false;
      }

   for( int i = squareRoot + 1; i <= n; i++ )
      if( primes[ i ] )
         numPrimes++;

   return numPrimes;
}
```

```
bool primes[ 1000000001 ];

int eratosthenes( int n )
{
   for( int i = 3; i <= number; i += 2 )
      primes[ i ] = true;

   int squareRoot = static_cast< int >( sqrt( static_cast< double >( n ) ) );
   int numPrimes = 0;

   for( int i = 3; i <= number; i += 2 )
      if( primes[ i ] )
      {
         numPrimes++;
         for( int j = 3 * i; j <= number; j += i + i )
            primes[ j ] = false;
      }

   for( int i = squareRoot + 1; i <= n; i++ )
      if( primes[ i ] )
         numPrimes++;

   return numPrimes;
}
```